Project1-CSE3027-Spring 2024

2022098068

Kim YoungTaek

1. Server Design

The server design comprises the following steps:

1. **Socket Creation**: Create sockets for IPv4 and TCP environments.
2. **Address Binding**: Assign IP addresses and port numbers to sockets for binding.
3. **Connection Reception**: Wait for incoming connection requests from clients.
4. **Client Connection Acceptance**: Accept client connection requests and create client sockets.
5. **Data Transmission**: Receive data from clients, process it, and return the content of requested files.
6. **Loop Execution**: The server continuously loops from the acceptance phase to closing the client socket, ensuring continuous reception of requests from clients.

Before opening the server, it receives the following arguments:

- If no port number is provided, an error occurs, and the program terminates.
- Upon successful reception of the port number, sockets for IPv4 and TCP environments are created. Then, IP addresses and port numbers are assigned to the sockets. Additionally, the socket's option is set using the setsockopt() function to ensure immediate termination of the server upon program termination.

Once the address binding is successfully completed, the server waits for client connections. Upon successful establishment of a connection, the server reads the client's request and processes it accordingly, including opening and returning the content of requested files. After completing these steps, the server closes the file and the client socket. Then, it accepts the client socket again and repeats the above steps until the user forcibly terminates the program.

2. Difficulties

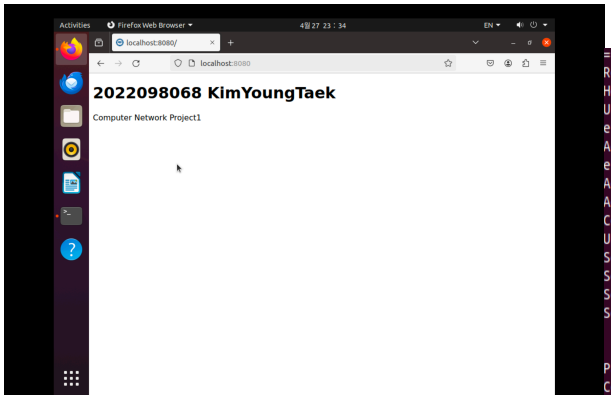
The main challenges I experienced while implementing the above code were the complexities of socket programming and the manual parsing of HTTP requests. Managing socket options and connections in C was particularly demanding, requiring precise error management and handling of exceptional situations. Additionally, extracting data from HTTP headers and understanding the format of incoming requests demanded a great deal of attention and was time-consuming. These factors combined made the overall task feel quite challenging.

3. Sample Output

```
taek@taek:~/Desktop/ComputerNetwork$ ./server 8080
```

After opening a server on port 8080 using the above commands, here are the respective output values.

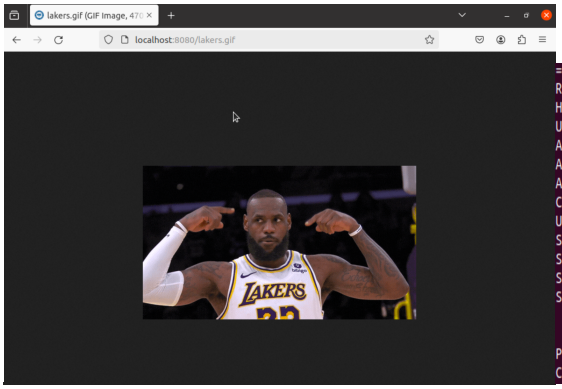
1) localhost:8080(index.html)




```
================================================
Request: GET / HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0) Gecko/20100101 Fir
efox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Path: /index.html
Content-Type: text/html
```

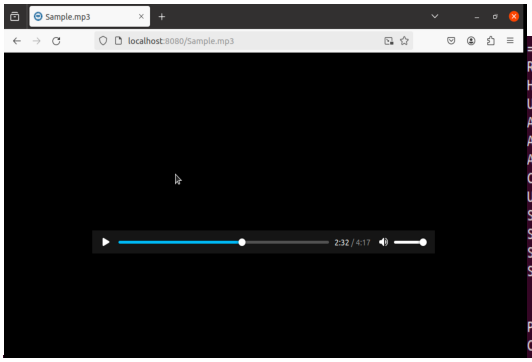
2) localhost:8080/lakers.gif



```
================================================
Request: GET /lakers.gif HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Path: /lakers.gif
Content-Type: image/gif
```

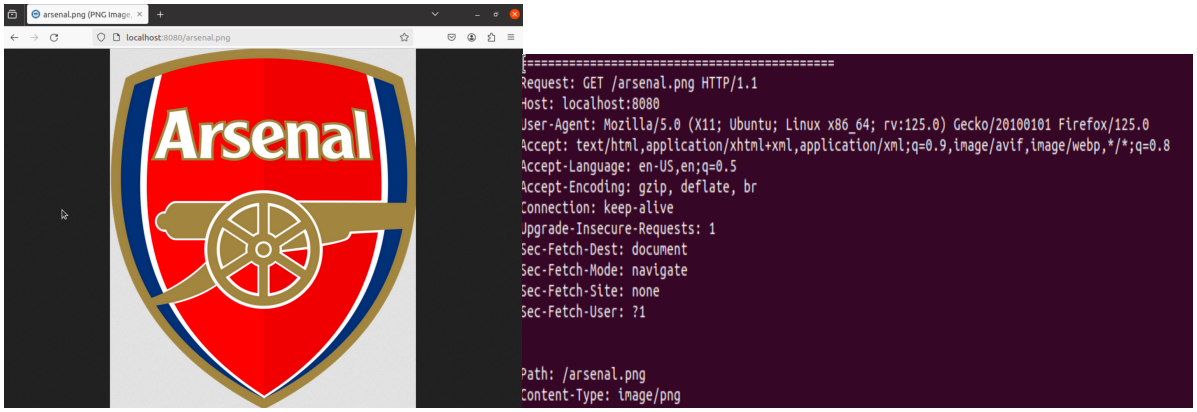
3) localhost:8080/Sample.mp3



```
================================================
Request: GET /Sample.mp3 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Path: /Sample.mp3
Content-Type: audio/mpeg
```

4) localhost:8080/arsenal.png



5) localhost:8080/proj1.pdf